Rocky Enterprise Linux 9.2 Manual Pages on command 'zipinfo.1'

C:\>man zipinfo.1

ZIPINFO(1) General Commands Manual ZIPINFO(1)

NAME

zipinfo - list detailed information about a ZIP archive

SYNOPSIS

zipinfo [-12smlvhMtTz] file[.zip] [file(s) ...] [-x xfile(s) ...]

unzip -Z [-12smlvhMtTz] file[.zip] [file(s) ...] [-x xfile(s) ...]

DESCRIPTION

zipinfo lists technical information about files in a ZIP archive, most commonly found on MS-DOS systems. Such information includes file access permissions, en? cryption status, type of compression, version and operating system or file system of compressing program, and the like. The default behavior (with no options) is to list single-line entries for each file in the archive, with header and trailer lines providing summary information for the entire archive. The format is a cross between Unix ``ls -l'' and ``unzip -v'' output. See DETAILED DESCRIPTION below.

Note that zipinfo is the same program as unzip (under Unix, a link to it); on some systems, however, zipinfo support may have been omitted when unzip was compiled.

ARGUMENTS

file[.zip]

Path of the ZIP archive(s). If the file specification is a wildcard, each matching file is processed in an order determined by the operating system (or file system). Only the filename can be a wildcard; the path itself can? not. Wildcard expressions are similar to Unix egrep(1) (regular) expres?

sions and may contain:

* matches a sequence of 0 or more characters

? matches exactly 1 character

[...] matches any single character found inside the brackets; ranges are specified by a beginning character, a hyphen, and an ending charac? ter. If an exclamation point or a caret (`!' or `^') follows the left bracket, then the range of characters within the brackets is complemented (that is, anything except the characters inside the brackets is considered a match). To specify a verbatim left bracket, the three-character sequence ``[]'' has to be used.

(Be sure to quote any character that might otherwise be interpreted or modi? fied by the operating system, particularly under Unix and VMS.) If no matches are found, the specification is assumed to be a literal filename; and if that also fails, the suffix .zip is appended. Note that self-ex? tracting ZIP files are supported, as with any other ZIP archive; just spec? ify the .exe suffix (if any) explicitly.

[file(s)]

An optional list of archive members to be processed, separated by spaces. (VMS versions compiled with VMSCLI defined must delimit files with commas instead.) Regular expressions (wildcards) may be used to match multiple members; see above. Again, be sure to quote expressions that would other? wise be expanded or modified by the operating system.

[-x xfile(s)]

An optional list of archive members to be excluded from processing.

OPTIONS

-1 list filenames only, one per line. This option excludes all others; head? ers, trailers and zipfile comments are never printed. It is intended for use in Unix shell scripts.

-2 list filenames only, one per line, but allow headers (-h), trailers (-t) and zipfile comments (-z), as well. This option may be useful in cases where the stored filenames are particularly long.

-s list zipfile info in short Unix ``ls -l'' format. This is the default be? havior; see below.

-m list zipfile info in medium Unix ``ls -l'' format. Identical to the -s out?
put, except that the compression factor, expressed as a percentage, is also
listed.

-l list zipfile info in long Unix ``ls -l'' format. As with -m except that the
compressed size (in bytes) is printed instead of the compression ratio.

-v list zipfile information in verbose, multi-page format.

-h list header line. The archive name, actual size (in bytes) and total number
of files is printed.

-M pipe all output through an internal pager similar to the Unix more(1) com?
mand. At the end of a screenful of output, zipinfo pauses with a
``--More--'' prompt; the next screenful may be viewed by pressing the Enter
(Return) key or the space bar. zipinfo can be terminated by pressing the
``q'' key and, on some systems, the Enter/Return key. Unlike Unix more(1),
there is no forward-searching or editing capability. Also, zipinfo doesn't
notice if long lines wrap at the edge of the screen, effectively resulting
in the printing of two or more lines and the likelihood that some text will
scroll off the top of the screen before being viewed. On some systems the
number of available lines on the screen is not detected, in which case zip?
info assumes the height is 24 lines.

-t list totals for files listed or for all files. The number of files listed,
their uncompressed and compressed total sizes , and their overall compres?
sion factor is printed; or, if only the totals line is being printed, the
values for the entire archive are given. The compressed total size does not
include the 12 additional header bytes of each encrypted entry. Note that
the total compressed (data) size will never match the actual zipfile size,
since the latter includes all of the internal zipfile headers in addition to
the compressed data.

-T print the file dates and times in a sortable decimal format (yymmdd.hhmmss).
The default date format is a more standard, human-readable version with ab?
breviated month names (see examples below).

-U [UNICODE_SUPPORT only] modify or disable UTF-8 handling. When UNICODE_SUP?
PORT is available, the option -U forces unzip to escape all non-ASCII char?
acters from UTF-8 coded filenames as ``#Uxxxx''. This option is mainly pro?

vided for debugging purpose when the fairly new UTF-8 support is suspected to mangle up extracted filenames.

The option -UU allows to entirely disable the recognition of UTF-8 encoded filenames. The handling of filename codings within unzip falls back to the behaviour of previous versions.

-z include the archive comment (if any) in the listing.

DETAILED DESCRIPTION

zipinfo has a number of modes, and its behavior can be rather difficult to fathom if one isn't familiar with Unix ls(1) (or even if one is). The default behavior is to list files in the following format:

-rw-rws--- 1.9 unx 2802 t- defX 11-Aug-91 13:48 perms.2660

The last three fields are the modification date and time of the file, and its name. The case of the filename is respected; thus files that come from MS-DOS PKZIP are always capitalized. If the file was zipped with a stored directory name, that is also displayed as part of the filename.

The second and third fields indicate that the file was zipped under Unix with ver? sion 1.9 of zip. Since it comes from Unix, the file permissions at the beginning of the line are printed in Unix format. The uncompressed file-size (2802 in this example) is the fourth field.

The fifth field consists of two characters, either of which may take on several values. The first character may be either `t' or `b', indicating that zip believes the file to be text or binary, respectively; but if the file is encrypted, zipinfo notes this fact by capitalizing the character (`T' or `B'). The second character may also take on four values, depending on whether there is an extended local header and/or an ``extra field'' associated with the file (fully explained in PKWare's APPNOTE.TXT, but basically analogous to pragmas in ANSI C--i.e., they pro? vide a standard way to include non-standard information in the archive). If nei? ther exists, the character will be a hyphen (`-'); if there is an extended local header but no extra field, `l'; if the reverse, `x'; and if both exist, `X'. Thus the file in this example is (probably) a text file, is not encrypted, and has nei? ther an extra field nor an extended local header associated with it. The example below, on the other hand, is an encrypted binary file with an extra field:

RWD,R,R 0.9 vms 168 Bx shrk 9-Aug-91 19:15 perms.0644

Extra fields are used for various purposes (see discussion of the -v option below) including the storage of VMS file attributes, which is presumably the case here. Note that the file attributes are listed in VMS format. Some other possibilities for the host operating system (which is actually a misnomer--host file system is more correct) include OS/2 or NT with High Performance File System (HPFS), MS-DOS, OS/2 or NT with File Allocation Table (FAT) file system, and Macintosh. These are denoted as follows:

```
-rw-a--     1.0 hpf    5358 Tl i4:3  4-Dec-91 11:33 longfilename.hpfs
-r--ahs     1.1 fat    4096 b- i4:2 14-Jul-91 12:58 EA DATA. SF
--w-------  1.0 mac   17357 bx i8:2  4-May-92 04:02 unzip.macr
```

File attributes in the first two cases are indicated in a Unix-like format, where the seven subfields indicate whether the file: (1) is a directory, (2) is readable (always true), (3) is writable, (4) is executable (guessed on the basis of the ex? tension--.exe, .com, .bat, .cmd and .btm files are assumed to be so), (5) has its archive bit set, (6) is hidden, and (7) is a system file. Interpretation of Macin? tosh file attributes is unreliable because some Macintosh archivers don't store any attributes in the archive.

Finally, the sixth field indicates the compression method and possible sub-method used. There are six methods known at present: storing (no compression), reducing, shrinking, imploding, tokenizing (never publicly released), and deflating. In ad? dition, there are four levels of reducing (1 through 4); four types of imploding (4K or 8K sliding dictionary, and 2 or 3 Shannon-Fano trees); and four levels of deflating (superfast, fast, normal, maximum compression). zipinfo represents these methods and their sub-methods as follows: stor; re:1, re:2, etc.; shrk; i4:2, i8:3, etc.; tokn; and defS, defF, defN, and defX.

The medium and long listings are almost identical to the short format except that they add information on the file's compression. The medium format lists the file's compression factor as a percentage indicating the amount of space that has been ``removed'':

```
-rw-rws---  1.5 unx    2802 t- 81% defX 11-Aug-91 13:48 perms.2660
```

In this example, the file has been compressed by more than a factor of five; the compressed data are only 19% of the original size. The long format gives the com? pressed file's size in bytes, instead:

-rw-rws--- 1.5 unx 2802 t- 538 defX 11-Aug-91 13:48 perms.2660

In contrast to the unzip listings, the compressed size figures in this listing for? mat denote the complete size of compressed data, including the 12 extra header bytes in case of encrypted entries.

Adding the -T option changes the file date and time to decimal format:

-rw-rws--- 1.5 unx 2802 t- 538 defX 910811.134804 perms.2660

Note that because of limitations in the MS-DOS format used to store file times, the seconds field is always rounded to the nearest even second. For Unix files this is expected to change in the next major releases of zip(1) and unzip.

In addition to individual file information, a default zipfile listing also includes header and trailer lines:

Archive: OS2.zip 5453 bytes 5 files

,,rw, 1.0 hpf 730 b- i4:3 26-Jun-92 23:40 Contents

,,rw, 1.0 hpf 3710 b- i4:3 26-Jun-92 23:33 makefile.os2

,,rw, 1.0 hpf 8753 b- i8:3 26-Jun-92 15:29 os2unzip.c

,,rw, 1.0 hpf 98 b- stor 21-Aug-91 15:34 unzip.def

,,rw, 1.0 hpf 95 b- stor 21-Aug-91 17:51 zipinfo.def

5 files, 13386 bytes uncompressed, 4951 bytes compressed: 63.0%

The header line gives the name of the archive, its total size, and the total number of files; the trailer gives the number of files listed, their total uncompressed size, and their total compressed size (not including any of zip's internal over? head). If, however, one or more file(s) are provided, the header and trailer lines are not listed. This behavior is also similar to that of Unix's ``ls -l''; it may be overridden by specifying the -h and -t options explicitly. In such a case the listing format must also be specified explicitly, since -h or -t (or both) in the absence of other options implies that ONLY the header or trailer line (or both) is listed. See the EXAMPLES section below for a semi-intelligible translation of this nonsense.

The verbose listing is mostly self-explanatory. It also lists file comments and the zipfile comment, if any, and the type and number of bytes in any stored extra fields. Currently known types of extra fields include PKWARE's authentication (``AV'') info; OS/2 extended attributes; VMS filesystem info, both PKWARE and Info-ZIP versions; Macintosh resource forks; Acorn/Archimedes SparkFS info; and so on.

(Note that in the case of OS/2 extended attributes--perhaps the most common use of zipfile extra fields--the size of the stored EAs as reported by zipinfo may not match the number given by OS/2's dir command: OS/2 always reports the number of bytes required in 16-bit format, whereas zipinfo always reports the 32-bit stor? age.)

Again, the compressed size figures of the individual entries include the 12 extra header bytes for encrypted entries. In contrast, the archive total compressed size and the average compression ratio shown in the summary bottom line are calculated without the extra 12 header bytes of encrypted entries.

ENVIRONMENT OPTIONS

Modifying zipinfo's default behavior via options placed in an environment variable can be a bit complicated to explain, due to zipinfo's attempts to handle various defaults in an intuitive, yet Unix-like, manner. (Try not to laugh.) Neverthe? less, there is some underlying logic. In brief, there are three ``priority lev? els'' of options: the default options; environment options, which can override or add to the defaults; and explicit options given by the user, which can override or add to either of the above.

The default listing format, as noted above, corresponds roughly to the "zipinfo -hst" command (except when individual zipfile members are specified). A user who prefers the long-listing format (-l) can make use of the zipinfo's environment variable to change this default:

Unix Bourne shell:

```
ZIPINFO=-l; export ZIPINFO
```

Unix C shell:

```
setenv ZIPINFO -l
```

OS/2 or MS-DOS:

```
set ZIPINFO=-l
```

VMS (quotes for lowercase):

```
define ZIPINFO_OPTS "-l"
```

If, in addition, the user dislikes the trailer line, zipinfo's concept of ``nega? tive options'' may be used to override the default inclusion of the line. This is accomplished by preceding the undesired option with one or more minuses: e.g., ``-l-t'' or ``--tl'', in this example. The first hyphen is the regular switch

character, but the one before the `t' is a minus sign. The dual use of hyphens may seem a little awkward, but it's reasonably intuitive nonetheless: simply ignore the first hyphen and go from there. It is also consistent with the behavior of the Unix command nice(1).

As suggested above, the default variable names are ZIPINFO_OPTS for VMS (where the symbol used to install zipinfo as a foreign command would otherwise be confused with the environment variable), and ZIPINFO for all other operating systems. For compatibility with zip(1), ZIPINFOOPT is also accepted (don't ask). If both ZIP? INFO and ZIPINFOOPT are defined, however, ZIPINFO takes precedence. unzip's diag? nostic option (-v with no zipfile name) can be used to check the values of all four possible unzip and zipinfo environment variables.

EXAMPLES

To get a basic, short-format listing of the complete contents of a ZIP archive storage.zip, with both header and totals lines, use only the archive name as an ar? gument to zipinfo:

zipinfo storage

To produce a basic, long-format listing (not verbose), including header and totals lines, use -l:

zipinfo -l storage

To list the complete contents of the archive without header and totals lines, ei? ther negate the -h and -t options or else specify the contents explicitly:

zipinfo --h-t storage

zipinfo storage *

(where the backslash is required only if the shell would otherwise expand the `*' wildcard, as in Unix when globbing is turned on--double quotes around the asterisk would have worked as well). To turn off the totals line by default, use the envi? ronment variable (C shell is assumed here):

setenv ZIPINFO --t

zipinfo storage

To get the full, short-format listing of the first example again, given that the environment variable is set as in the previous example, it is necessary to specify the -s option explicitly, since the -t option by itself implies that ONLY the footer line is to be printed:

setenv ZIPINFO --t

zipinfo -t storage [only totals line]

zipinfo -st storage [full listing]

The -s option, like -m and -l, includes headers and footers by default, unless oth? erwise specified. Since the environment variable specified no footers and that has a higher precedence than the default behavior of -s, an explicit -t option was nec? essary to produce the full listing. Nothing was indicated about the header, how? ever, so the -s option was sufficient. Note that both the -h and -t options, when used by themselves or with each other, override any default listing of member files; only the header and/or footer are printed. This behavior is useful when zipinfo is used with a wildcard zipfile specification; the contents of all zipfiles are then summarized with a single command.

To list information on a single file within the archive, in medium format, specify the filename explicitly:

zipinfo -m storage unshrink.c

The specification of any member file, as in this example, will override the default header and totals lines; only the single line of information about the requested file will be printed. This is intuitively what one would expect when requesting information about a single file. For multiple files, it is often useful to know the total compressed and uncompressed size; in such cases -t may be specified ex? plicitly:

zipinfo -mt storage "*.[ch]" Mak*

To get maximal information about the ZIP archive, use the verbose option. It is usually wise to pipe the output into a filter such as Unix more(1) if the operating system allows it:

zipinfo -v storage | more

Finally, to see the most recently modified files in the archive, use the -T option in conjunction with an external sorting utility such as Unix sort(1) (and sed(1) as well, in this example):

zipinfo -T storage | sort -nr -k 7 | sed 15q

The -nr option to sort(1) tells it to sort numerically in reverse order rather than in textual order, and the -k 7 option tells it to sort on the seventh field. This assumes the default short-listing format; if -m or -l is used, the proper sort(1)

option would be -k 8. Older versions of sort(1) do not support the -k option, but you can use the traditional + option instead, e.g., +6 instead of -k 7. The sed(1) command filters out all but the first 15 lines of the listing. Future releases of zipinfo may incorporate date/time and filename sorting as built-in options.

TIPS

The author finds it convenient to define an alias ii for zipinfo on systems that allow aliases (or, on other systems, copy/rename the executable, create a link or create a command file with the name ii). The ii usage parallels the common ll alias for long listings in Unix, and the similarity between the outputs of the two commands was intentional.

BUGS

As with unzip, zipinfo's -M (``more'') option is overly simplistic in its handling of screen output; as noted above, it fails to detect the wrapping of long lines and may thereby cause lines at the top of the screen to be scrolled off before being read. zipinfo should detect and treat each occurrence of line-wrap as one addi? tional line printed. This requires knowledge of the screen's width as well as its height. In addition, zipinfo should detect the true screen geometry on all sys? tems.

zipinfo's listing-format behavior is unnecessarily complex and should be simpli? fied. (This is not to say that it will be.)

SEE ALSO

ls(1), funzip(1), unzip(1), unzipsfx(1), zip(1), zipcloak(1), zipnote(1), zip? split(1)

URL

The Info-ZIP home page is currently at

http://www.info-zip.org/pub/infozip/

or

ftp://ftp.info-zip.org/pub/infozip/ .

AUTHOR

Greg ``Cave Newt'' Roelofs. ZipInfo contains pattern-matching code by Mark Adler and fixes/improvements by many others. Please refer to the CONTRIBS file in the UnZip source distribution for a more complete list.